NEtwork of Research Infrastructures for European Seismology

Report
D4- Implementation of accelerometric parameters computation and exchange:

PART 1-Computation software

Activity:	Improving accelerometric data dissemination
Activity number:	NA5 Task B
Deliverable:	Accelerometric parameters computation and exchange
Deliverable number:	D4-Updated
Date:	February 2010
Responsible activity leader:	Antoni Roca
Responsible participant:	IGC, Institut Geològic de Catalunya IST, Instituto Superior Técnico
Authors:	Albert Marsal, Teresa Susagna, Xavier Goula and Carlos Sousa Oliveira

Sixth Framework Programme
EC project number: 026130

Summary

The proposal of the Task B inside the NA5 was: *the definition and implementation of parametric data exchange procedures.* The participants on this task are: *IGC, LGIT, ITSAK, ETHZ, KOERI, EMSC, IST and ETHZ*

Concretely, the main objectives are:

- Definition of the parametric data to be exchanged and the procedures to compute them.
- Implementation of software at networks.
- To promote the exchange of Pseudo Spectral Acceleration (PSA) and Pseudo Spectral Velocity (PSV), extremely useful to constrain shakemaps (→ JRA3), make comparisons with community intensity maps (→ NA7) and validate attenuation laws.
- To initiate the collection and distribution of parametric data through the portal (→NA7, TA1).

This task must generate the following objects:

- The software to obtain from every record the parametric data.

  At this point it is important to remark that the aim of this project is not to provide a software tool to compute accelerometric parameters in a customizable way. It must be taken into account that the main idea is to disseminate a common, homogenous and automatic way to compute that parameters. Therefore, the processing procedures to compute parameters will be always the same with any event, whatever its magnitude, source, etc.

  Neither is it a goal of the project to disseminate time-histories of the events (filtered acceleration, velocity and displacement) although they are calculated.

- The parametric files and the event information files which will content the metadata needed to perform the searches of records and, in a future, to constrain shakemaps, validate attenuation laws, etc.

In order to choose the adequate procedures to be implemented in the parameters computation software, a miniBenchmark exercise was proposed. Then, a common procedure for parameters computation was studied and implemented. The parameters computation software was distributed to all participants and their comments were taken into account to generate new versions.

After the NERIES meeting of March 2008 in Grenoble some improvements has been introduced. Recommendations expressed by external reviewers have been considered (October 2008, update).

After a comparison of parameters from ITACA (INGV) computed with ParamaccV8 an error was detected in Housner Intensity routine and corrected in a new version ParamaccV9 (February 2010, updated).

- *Task B deliverables:*

#	Deliverable title	Date	Nature	Task
D2	Specifications for PSA and PSV Definition and computation of parametric data	9	R, Software	B
D4	Implementation of accelerometric parameters computation and exchange	18	R, Web	B

This deliverable D4 is composed by two parts:

Part 1: corresponds to the presentation of the last version of the parameters computation software. It contains a general description of user interfaces, computation module and visualization module; furthermore, there are 3 annexes containing source code of the software and Paramacc and RegisterVisor user manuals.

Part 2: contains the results of each participant in reference to the application of the parameters computation software to its set of files.

PART 1-The parameters computation software

Matlab platform is the language selected to implement the common procedure of parameter computation due to its wide scientific use and the routines that the signal processing toolbox offers. In addition, Matlab allows compiling the scripts into self-contained programs for different platforms (Windows, UNIX, Linux…).

A compilation under Windows platform and the source code have been distributed among the NA5 partners. The name of the software is **paramacc**.

In parallel, a second program to visualize the calculated parameters has been developed. It is briefly mentioned in the following pages and its name is **registervisor**.

This issue of D4 (October 2008) corresponds to the version 8 of both softwares.

1-ParamaccV9, general structure

The basic elements of the computation software are shown in the scheme of the Figure 1.

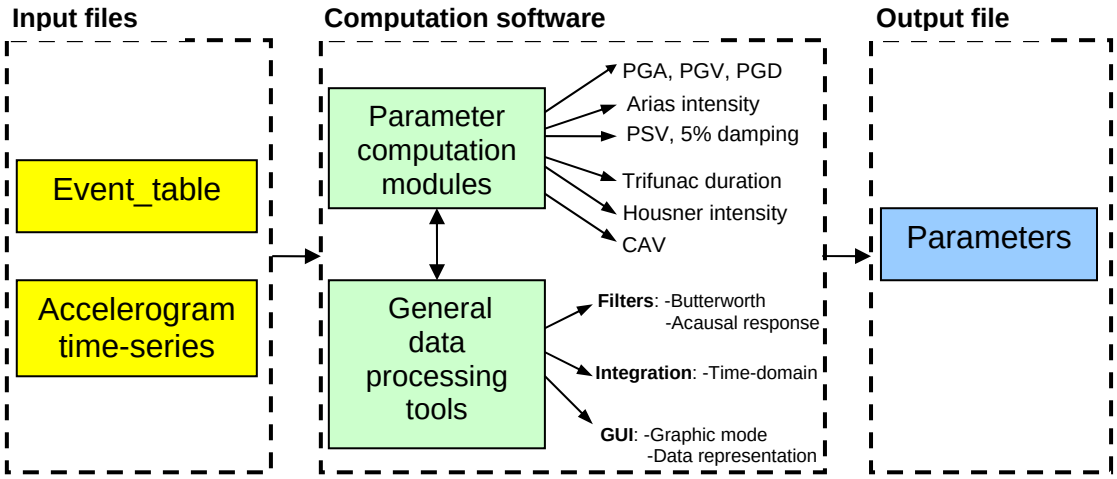


Figure 1. Scheme of the Matlab software paramaccV9 for the computation of parameters

Basically, the computation software is composed of three main parts:

-a **user interface** that allows selecting input files to read the data to be processed.

-the **parameters computation modules**, which calculate each output parameter individually with the help of filters, time-domain integrations and many other general purpose tools.

-the **module to write** parameters in a formatted file and, additionally, to **visualize** computed data.

See Appendix A for complete list of the software and appendix B for the user manual of **ParamaccV9.**

1.1-User interface

This graphic module allows the user to interact with the program in order to:

a) Read the input file (*event_table*).

With the file browser it is possible to open any file in any directory. See Figure 2 to get a better idea of the user window.

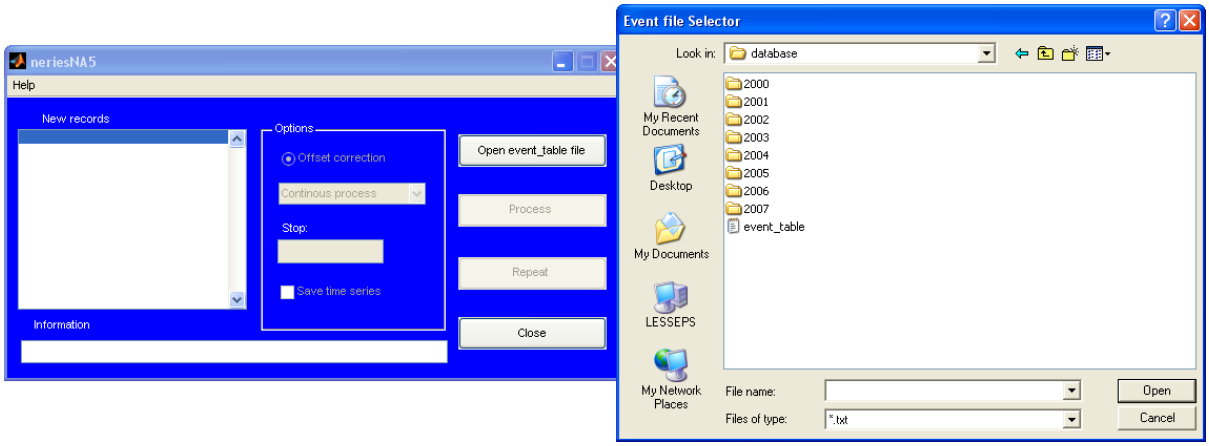


Figure 2. User interface with the *event_table* file browser opened

Before executing the program, it is absolutely necessary to have an *event_table* already created. Each partner must generate an *event_table* file with the records of its own area of interest. The format of the event file must be the following:

Table 1. Example of event_table.txt

ID			**Date**	**time_event**	**time_record**	**Lon**	**Lat**	**Depth**
20000211_0000012			2000/02/11	16:16:29.8	16:16:23.0	2.06	42.48	9
Mw	**Ml**	**Mb**	**network**	**record**				
?	2.1	?	IGC	20000211.161630.LLIR-IGC.01.HNE.asc				

Notes:
-It is important to remark that each field must be filled with a character different from blank spaces.
-The record name must have 35 characters including points, dashes and the extension of the file.
-The ID is an unified identifier that will be furnished by EMSC-CSEM.
-Time-series of accelerometric data must be saved classified by year, month and day and in the same folder that the *event_table*:

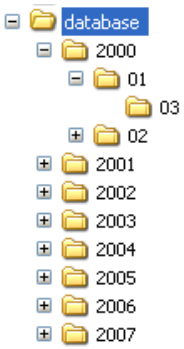


Figure 3. Data structure. In this example, *event_table* should be placed in the "database" folder.

b) Start processing parameters.

Once the input file has been read, the user can start to process the records.

1.2- Parameters computation modules

It is considered that parameters computation modules are a set of specific functions that focus on the computation of each parameter. Consequently, there is one function for each parameter to calculate. The next flowchart shows the steps for the computation process.

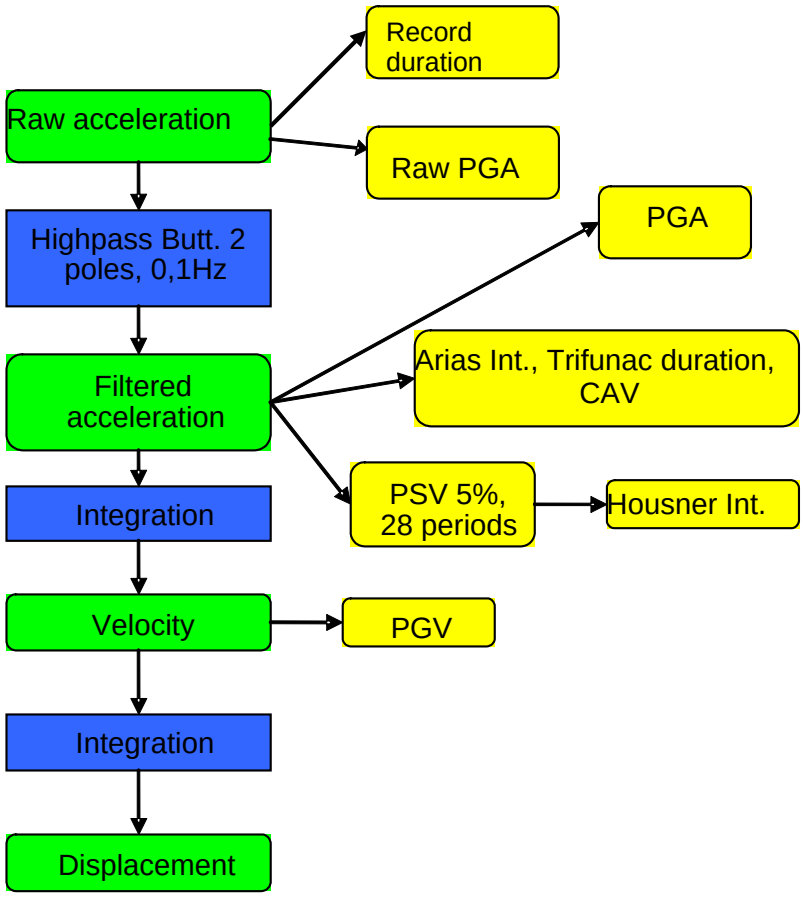


Figure 4. Flowchart for the parameters computation

Here is a more detailed explanation about the function which computes each parameter (see appendix A):

- Raw acceleration: acceleration time-history in cm/s^2, base line corrected. It is supposed that the user should remove the offset of the record before processing it. In spite of this, the software allows (optional) an automatic base line correction for raw acceleration records by one degree polynomial approximation fitted in a least squares sense.

- Record duration: duration of raw acceleration record (in seconds).

- Raw PGA (cm/s^2): PGA (peak ground acceleration) from raw acceleration record.

- Highpass filter (acausal): Butterworth IIR highpass filter of two poles is implemented. To maintain the homogeneity and to avoid being too restrictive, the cut-off frequency is 0,1Hz for all records, taking into account their variety and instruments resolution. Filtering is applied again in the opposite time

direction in order to avoid phase distortion. Data padding has been introduced to avoid low frequency distortion. A number of zeros equivalent to 5% of the time duration has been added, both at the beginning and at the end of signal.

- Filtered acceleration: raw acceleration after the application of the previously defined filter.

- PGA (cm/s^2): PGA from filtered record. It is directly obtained from the maximum value of the filtered acceleration time-history.

- AI (cm/s): Arias intensity. A specific function detailed in “Int_arias.m” according the next expression:

$$I.A. = \frac{\pi}{2 \cdot g} \int_0^{\infty} a^2(t) dt$$

- Trifunac duration (s): Trifunac duration is the time interval between the 5% and 95% of a Husid plot, detailed in “Trifunac.m”:

$$Husid(t) = \frac{\int_0^{t} a^2(t) dt}{\int_0^{\infty} a^2(t) dt}$$

- CAV (cm/s): Cumulative Absolute Velocity. A specific function detailed in “Cav.m” according to the next expression:

$$CAV = \int_0^{\infty} |a(t)| dt$$

- PSV (5%) (pseudovelocity) (cm/s) for 28 frequencies logarithmically equally spaced (from 0.15Hz to 39Hz) (frequencies: 0.15, 0.19, 0.23, 0.28, 0.34, 0.42, 0.52, 0.64, 0.78, 0.96, 1.18, 1.45, 1.78, 2.19, 2.69, 3.31, 4.06, 4.99, 6.13, 7.53, 9.25, 11.37, 13.96, 17.15, 21.07, 25.89, 31.80, 39.07 Hz). The first 6 frequencies PSV values are not offered in case of PGA<0.01g or PGV<1cm/s.

The function is detailed in “Psv.m”.

- Housner intensity or response spectrum intensity (cm). A specific function detailed in “Housner.m” according to the next expression:

$$I_{Housner}(\xi) = \int_{0,1}^{2,5} PSV(\xi, T) dT \,, \qquad \text{with } \xi = 5\%$$

-Integration: via the trapezoidal method (time domain) to obtain both velocity and displacement time-histories. After filtering the first time to remove noise from filtered acceleration record, any other filter is applied.

-Velocity time history (cm/s): integrated filtered acceleration time history.

- PGV (peak ground velocity) in cm/s. It is directly obtained from the maximum value of the calculated velocity time-history.

- Displacement time history (cm): integrated velocity time history.

1.3-Writing/Visualizing module

While the output parameters are calculated, a file called “parameters_table.txt” is updated. This file is placed in the same folder that the *event_table*. It is important to know that this file will be generated automatically by the program and, if it was already created, it will be updated with the new events reported in the *event_table* since the last execution of the program.

The next table shows the format of the “parameters_table.txt”. Note that all the values are placed in one row for each record.

Table 2. Example of parameters_table.txt

Record	**rec. duration.**	**PGA_uncor.**	**PGA**	**Arias intensity**
20080101.000000.KOER-PPP.01.HGV.asc	30.036000	735.393877	733.029465	652.90104170

Trifunac d. CAV PGV PSV(at 28 freq.) damping=5%

12.880000 2138.729872 37.335376 32.054378 26.574545 18.436520 16.889276 13.165072 18.007773 31.264875 63.818260 37.773259 37.081048 31.696547 26.919527 34.960381 43.099102 43.998432 66.653943 83.237890 73.487635 57.535370 35.168558 23.043350 13.348033 9.029967 7.373125 6.458809 5.108710 3.453048 2.835683

Housner intensity

65.078088

The visualization module is optional and operates with the parameters calculated for the last record. The goal of this possibility is to check the processing by the visualization of acceleration, velocity and displacement time-histories and response spectrum (figure 5).

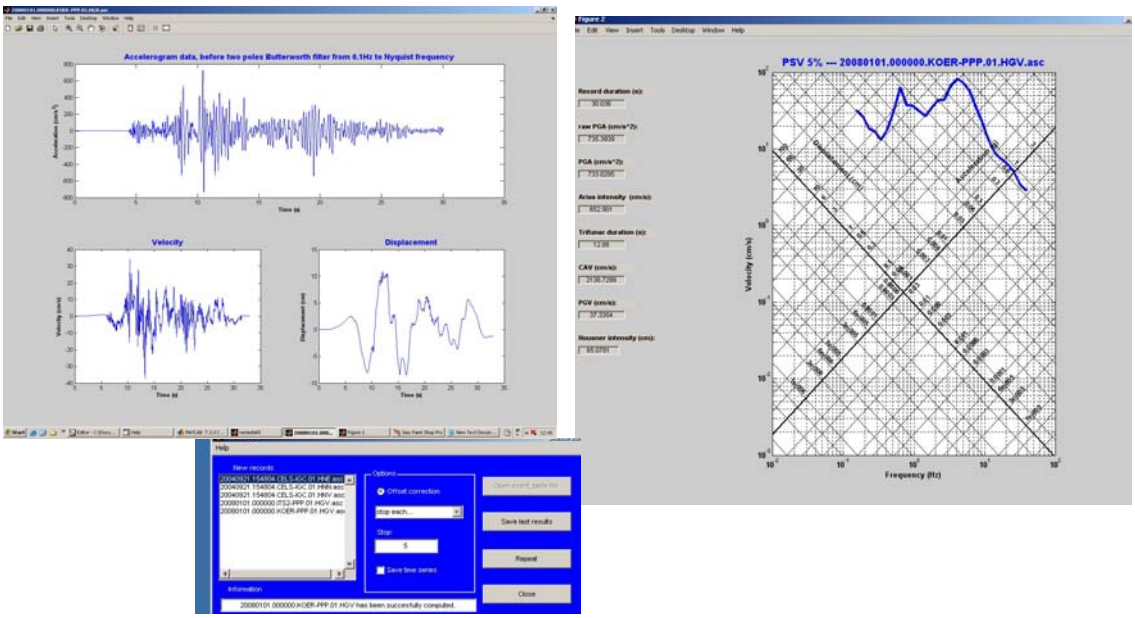


Figure 5. Visualization mode

Refer to “soft_paramacc.pdf” for a more detailed explanation about user instructions.

2-RegistervisorV9, general structure

The basic elements of the visualization software are represented in the next scheme in Figure 6.

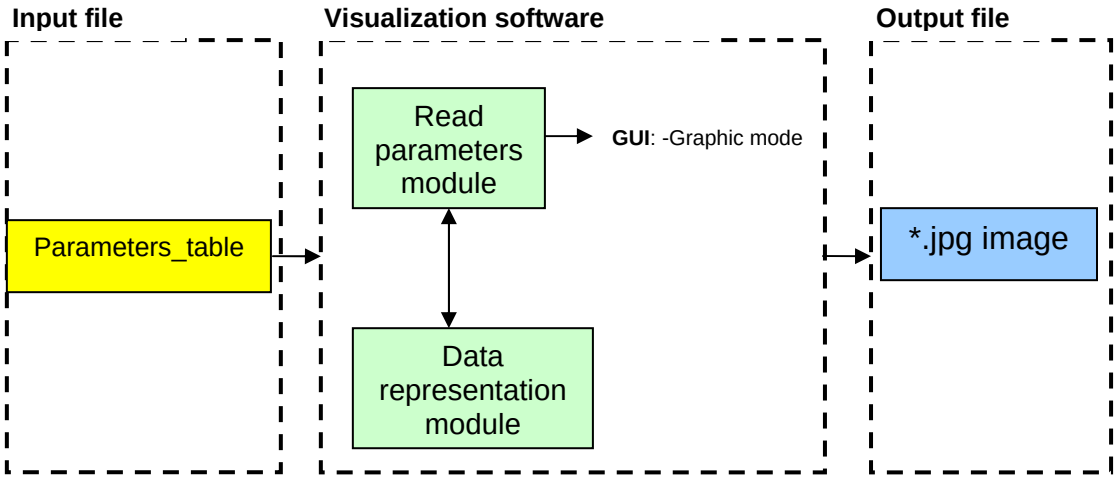


Figura 6. Scheme of the Matlab software registervisor for the visualization of parameters

Registervisor reads the *parameters_table* previously generated and the raw acceleration time history, and creates an image with the value of each parameter and a plot with raw acceleration time history and PSV 5%. This is only a tool for checking the output parameters and it doesn't manipulate anything by itself.

The program has a file browser to open any file in any directory. Only files generated by paramaccV9 will be correctly read. The records available to be represented will be showed in a window. The next figure 7 shows an example of an output file and the graphic mode of the program:

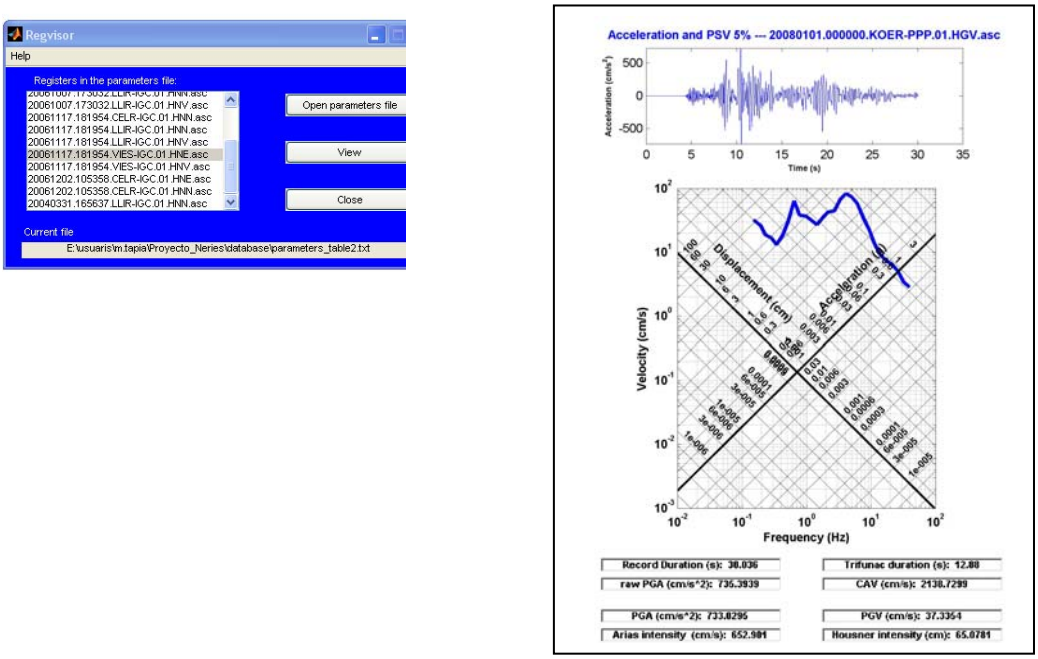


Figure 7. Registervisor software (left) and output file (right).

All generated images are saved and placed in the folder where the *parameters_table* is located.

See Appendix C for the user manual of registervisorV9.

Appendix A: Software source code related to parameters computation modules

-**ParamaccV9**: is the initial part of the program. It manages the *user interface* and enables the user to start the processing of the records.
-**Read_event**: it reads the event_table file and the parameters_table file and it compares both to establish the new records to be computed.
-**Mainprog**: according to the information received from “Read_event”, it starts the signal processing and writes the output parameters to the parameters file. It calls Central_defV2.
-**Central_defV2**: according to “mainprog”, it reads and processes each accelerometric record. This action includes the call of every computing parameters routines.

ParamaccV9

```
function
paramaccV9(action,ip_file,op_file,path,option1,option2,option3)

%%%%%%%%Seismic parameters calculation software into NERIES
project%%%%%%%%
%
%
paramaccV9('action','ip_file','op_file','path','option1','option2','op
tion3')
%    executes the software in an automatic mode.
%
%       action: no minds, type anythig.
%
%       ip_file: events input file with extension.
%
%       op_file: parameters output file with extension.
%
%       path: complet path of input file, output file and time series.
%
%       option1: 'Default' to run software with default options, i.e.
with offset correction
%                and not saving time series.
%                'Custom' to run software according to option2 and
option3.
%
%       option2: '1' to apply offset correction.
%                '0' to not apply offset correction.
%
%       option3: '1' to save time series.
%                '0' to not save time serie.
%
%   paramaccV9 to run software manually.
%
%   Examples:
%
paramaccV9('','events_table.txt','parameters_table.txt','C:\Documents
and Settings','Default');
%
%
paramaccV9('','events_table.txt','parameters_table.txt','C:\Documents
and Settings','Custom','1','1');
```

```
global ctrl;
global iline;
global iplines;
global odata;
global pathname;
global cont0;
global fid20;
global nproc;
global neweve;
global firstime;
global direct_name;

if(nargin<1)
    f=openfig('principal.fig');
    ctrl=guihandles(f);
    nproc=0;
    firstime=0;
    direct_name='';
    return
end
if(nargin==1)
    if (strcmp(action,'open'))
        [fid20,elines,plines,pathname,cont0,neweve]=read_event(ctrl);
        if(elines > plines)
            iline=1;
            iplines=1;
        else
            iline=0;
        end
    end
    if (strcmp(action,'process'))
        if(nproc==1)
            wid=size(odata);
            for n=1:wid(1)
                fprintf(fid20,'%s %.6f %.6f %.6f %.8f %.6f %.6f %.6f
%.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f
%.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f
%.6f\n', cont0(neweve(iplines+n-1),:),
odata(n,1),odata(n,2),odata(n,3),odata(n,4),odata(n,5),odata(n,6),odat
a(n,7),odata(n,8),odata(n,9),odata(n,10),odata(n,11),odata(n,12),odata
(n,13),odata(n,14),odata(n,15),odata(n,16),odata(n,17),odata(n,18),oda
ta(n,19),odata(n,20),odata(n,21),odata(n,22),odata(n,23),odata(n,24),o
data(n,25),odata(n,26),odata(n,27),odata(n,28),odata(n,29),odata(n,30)
,odata(n,31),odata(n,32),odata(n,33),odata(n,34),odata(n,35),odata(n,3
6));
            end

        end
        iplines=iline;
        set(ctrl.pb1,'Enable','off');
        set(ctrl.pb2,'Enable','on');
        set(ctrl.pb3,'Enable','off');
        set(ctrl.pb4,'Enable','off');
        if (get(ctrl.pop,'Value')==1)
            [newlines odata firstime
direct_name]=mainprog(inf,get(ctrl.opcio,'Value'),fid20,iline,pathname
,cont0,ctrl,neweve,firstime,direct_name);
            iline=newlines;
            nproc=0;
            set(ctrl.pb1,'Enable','on');
            set(ctrl.pb2,'Enable','on');
```

```
            set(ctrl.pb3,'Enable','on');
            set(ctrl.pb4,'Enable','off');
        else
            nstops=str2num(get(ctrl.nstop,'String'));
            if ((nstops > 0)&(nstops <= (length(neweve)-iline+1)))
                [newlines odata firstime
direct_name]=mainprog(nstops,get(ctrl.opcio,'Value'),fid20,iline,pathn
ame,cont0,ctrl,neweve,firstime,direct_name);
                iline=newlines;
                nproc=1;
                set(ctrl.pb1,'Enable','off');
                set(ctrl.pb2,'Enable','on');
                set(ctrl.pb3,'Enable','on');
                set(ctrl.pb4,'Enable','on');
            else
                if(length(neweve)==iline-1)
                    set(ctrl.action,'String','All events have been
processed. Program finished');
                    set(ctrl.pb1,'Enable','on');
                    set(ctrl.pb2,'Enable','on');
                    set(ctrl.pb3,'Enable','on');
                    set(ctrl.pb4,'Enable','off');
                    nproc=0;
                else
                    errordlg('Number of stops must be correctly
introduced');
                    set(ctrl.pb1,'Enable','off');
                    set(ctrl.pb2,'Enable','on');
                    set(ctrl.pb3,'Enable','on');
                    set(ctrl.pb4,'Enable','on');
                    nproc=0;
                end
            end
        end
    end
    if (strcmp(action,'repeat'))
        iline=iplines;
        nproc=0;
        paramaccV9('process');
    end
    if (strcmp(action,'close'))
        close('all');
        fclose('all');
    end
end

if (nargin==5)|(nargin==6)|(nargin==7)
    f=openfig('principal.fig');
    ctrl=guihandles(f);
    nproc=0;
    firstime=0;
    pathname=path;

[fid20,elines,plines,pathname,cont0,neweve]=read_event(ctrl,ip_file,op
_file,pathname);
    if(elines > plines)
        iline=1;
        iplines=1;
    else
        iline=0;
    end
```

```
    if (strcmp(option1,'Custom'))
         if (exist('option2'))
            if (strcmp(option2,'1'))
                set(ctrl.opcio,'Value',1);
            else
                if (strcmp(option2,'0'))
                    set(ctrl.opcio,'Value',0);
                else
                    msgbox('Unknown input argument','Info','error');
                end
            end
        end
        if (exist('option3'))
            if (strcmp(option3,'1'))
                set(ctrl.ck,'Value',1);
                direct_name=path;
            else
                if (strcmp(option3,'0'))
                    set(ctrl.ck,'Value',0);
                else
                    msgbox('Unknown input argument','Info','error');
                end
            end
        end
    else
        if (strcmp(option1,'Default'))
            set(ctrl.opcio,'Value',1);
            set(ctrl.ck,'Value',0);
        else
            msgbox('Unknown input argument','Info','error');
        end
    end
    if(fid20 > -1)
        paramaccV9('process');
        close(f);
        fclose('all');
        clear('all');
    end
end

if (nargin == 2)|(nargin == 3)|(nargin==4)
    msgbox('Input arguments error','Info','error');
end
```

read_event.m

```
function
[fid20,elines,plines,pathname,cont0,neweve]=read_event(ctrl,ip_file,op
_file,pathname)

%----------Function to recognize new events--------------------------
----%

%-----------------Select an events file-----------------------------
----%
if (nargin==1)
    [filename, pathname] =
uigetfile({'*.txt';'*.dat';'*.m';'*.*'},'Event file Selector');
    if(filename==0)
        fid20=-1;
```

```
        elines=0;
        plines=0;
        pathname='';
        cont0=0;
        neweve='';
        return;
    end
    clear('neweve');
    set(ctrl.lb,'String','');
    set(ctrl.action,'String','NA5-TaskB. Accelerometric parameter
computation');

%--------------------Create parameters file--------------------------
---%

    eventfile=cat(2,pathname,filename);
    paramfile='parameters_table.txt';
    paramfile=[pathname paramfile];
end

%------------------Open preselected event file and paramters file-----
---%
if (nargin==4)
    pathname=cat(2,pathname,'/');
    eventfile=cat(2,pathname,ip_file);
    paramfile=cat(2,pathname,op_file);
end
%-----------Open event file and parameters file----------------------
---%

fid10=fopen(eventfile,'r');
if(fid10==-1)
    fid20=-1;
    elines=0;
    plines=0;
    pathname='';
    cont0=0;
    neweve='';
    errordlg('Error opening the event file','File error');
    set(ctrl.action,'String','Can´t open event file.');
    return;
end
fid20=fopen(paramfile,'a+');
if(fid20==-1)
    fclose(fid10);
    elines=0;
    plines=0;
    pathname='';
    cont0=0;
    neweve='';
    errordlg('Error opening the parameters file','File error');
    set(ctrl.action,'String','Can´t open parameters file.');
    return;
end
%--------------Reading the records name from event file---------------
----%

elines=0;
for k=1:11
    fscanf(fid10,'%s',[1 1]);
```

```
end
year=fscanf(fid10,'%4s', [1 1]);
month=fscanf(fid10,'%2s', [1 1]);
day=fscanf(fid10,'%2s', [1 1]);
restname=fscanf(fid10,'%27s\n', [1 1]);
archivo1=[year month day restname];
while (isempty(archivo1)~=1)
    cont0(elines+1,:)=archivo1;
    for k=1:11
        fscanf(fid10,'%s',[1 1]);
    end
    year=fscanf(fid10,'%4s', [1 1]);
    month=fscanf(fid10,'%2s', [1 1]);
    day=fscanf(fid10,'%2s', [1 1]);
    restname=fscanf(fid10,'%27s\n', [1 1]);
    archivo1=[year month day restname];
    if ((length(archivo1)~=35)&(isempty(archivo1)~=1))
        fclose('all');
        elines=0;
        plines=0;
        pathname='';
        cont0=0;
        neweve='';
        errordlg('Wrong event file','File error');
        set(ctrl.action,'String','Wrong event file.');
        return;
    end
    elines=elines+1;
end

%---------Reading the records name from parameters file---------------
----%

fseek(fid20,0,-1);
tline=fgets(fid20);
plines=0;
while (tline~=-1)
    cont1(plines+1,:)=[tline(1:35)];
    tline=fgets(fid20);
    plines=plines+1;
end

%-------------------------------------------------------------------------
----%
if(exist('cont0'))
    lengthev=size(cont0);
else
    lengthev=0;
end
if(exist('cont1'))
    lengthpa=size(cont1);
else
    lengthpa=0;
end

%--------------Looking for new registers to treat--------------------
----%

index=1;
if(plines > 0)&(elines > 0)
```

```
    for (n=1:lengthev(1))
        for (r=1:lengthpa(1))
            if(strcmp(cont0(n,:),cont1(r,:))==1)
                break;
            elseif
((strcmp(cont0(n,:),cont1(r,:))==0)&(r==lengthpa(1)))
                neweve(index)=n;
                index=index+1;
            end
        end
    end
else
     for (n=1:lengthev(1))
         neweve(index)=n;
         index=index+1;
     end
end

%--------------Button enabling for the menu--------------------------
----%

set(ctrl.opcio,'Enable','on');
set(ctrl.pb3,'Enable','on');
set(ctrl.lb,'Enable','on');
set(ctrl.ck,'Enable','on');
if(exist('neweve'))
    for(i=1:length(neweve))
        prev=get(ctrl.lb,'String');
        set(ctrl.lb,'String',[prev;cont0((neweve(i)),:)]);
    end
else
    neweve=[];
end
set(ctrl.pop,'Enable','on');
set(ctrl.nstop,'Enable','on');

fclose(fid10);
```

Mainprog.m

```
function [newlines odata firstime
direct_name]=mainprog(stop,ofcl,fid20,iline,pathname,cont0,ctrl,neweve
,firstime,direct_name)

%----------Function to call the parameters calculation---------------
----%

clnoff=ofcl;
if(stop==1)
    pok=1;
else
    pok=0;
end

%--------Save time-series?-------------------------------------------
----%

if((get(ctrl.ck,'Value')==1)&&(firstime==0))
    firstime=1;
    if (strcmp(direct_name,''))
```

```
        direct_name=uigetdir('dialog_title','Select output time series
files destination');
    end
    if(direct_name==0)
        odata='';
        newlines=iline;
        firstime=0;
        return;
    end
else
    if(get(ctrl.ck,'Value')==0)
        firstime=0;
        direct_name=0;
    end
end

%--------Are all events already done?-------------------------------
----%

fseek(fid20,0,1);
if((iline-1==length(neweve))|(iline==0))
    set(ctrl.action,'String','All events have been treated. Program
finished');
    newlines=length(neweve)+1;
    odata='';
    return;
end

%----------Read the first new event-----------------------------------
----%

archivo1path=[pathname cont0(neweve(iline),1:4) '/'
cont0(neweve(iline),5:6) '/' cont0(neweve(iline),7:8) '/'
cont0(neweve(iline),:)];

%----------Calculate the first event parameters----------------------
----%

if (exist(archivo1path)==0)
    fclose('all');
    errordlg('Input file name or file path error.');
    set(ctrl.action,'String','Input file error.');
    newlines=0;
    odata='';
    set(ctrl.pb4,'Enable','on');
    return;
end
set(ctrl.action,'String',['Computing parameters of '
cont0(neweve(iline),1:31)]);
pause(0.1);
[SD,PGAu,PGAc,IA,TD,CAV,PGV,PSV,IH]=Central_defV2(archivo1path,clnoff,
pok,firstime,direct_name);

%-----------Put parameters in an intermediate variable---------------
----%

odata(1,:)=[SD PGAu PGAc IA TD CAV PGV PSV IH];
if(stop==inf)
    fprintf(fid20,'%s %.6f %.6f %.6f %.8f %.6f %.6f %.6f %.6f %.6f
%.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f
```

```
%.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f\n',
cont0(neweve(1),:),
odata(1,1),odata(1,2),odata(1,3),odata(1,4),odata(1,5),odata(1,6),odat
a(1,7),odata(1,8),odata(1,9),odata(1,10),odata(1,11),odata(1,12),odata
(1,13),odata(1,14),odata(1,15),odata(1,16),odata(1,17),odata(1,18),oda
ta(1,19),odata(1,20),odata(1,21),odata(1,22),odata(1,23),odata(1,24),o
data(1,25),odata(1,26),odata(1,27),odata(1,28),odata(1,29),odata(1,30)
,odata(1,31),odata(1,32),odata(1,33),odata(1,34),odata(1,35),odata(1,3
6));
end
if(pok==1)
    set(ctrl.action,'String',[cont0(neweve(iline),1:31) ' has been
succesfully computed.']);
    newlines=iline+1;
    if(iline==length(neweve))
        set(ctrl.pb3,'String','Save last results');
        msgbox('Press "save last results" button to save new data
parameters.','Info','warn');
    else
        msgbox('Press "process" button to follow processing and save
new data parameters.','Info','warn');
    end
    return;
end

%----------Calculate parameters for the rest of the events-----------
----%

for j=(iline+1):length(neweve)
      archivo1path=[pathname cont0(neweve(j),1:4) '/'
cont0(neweve(j),5:6) '/' cont0(neweve(j),7:8) '/' cont0(neweve(j),:)];
      if (exist(archivo1path)==0)
         fclose('all');
         errordlg('Input file name or file path error.');
         set(ctrl.action,'String','Input file error.');
         newlines=0;
         return;
      end
      if(rem(j-iline+1,stop)==0)
          pok=1;
      else
          pok=0;
      end
      set(ctrl.action,'String',['Computing parameters of '
cont0(neweve(j),1:31)]);
      pause(0.1);

[SD,PGAu,PGAc,IA,TD,CAV,PGV,PSV,IH]=Central_defV2(archivo1path,clnoff,
pok,firstime,direct_name);
      odata(j-iline+1,:)=[SD,PGAu PGAc IA TD CAV PGV PSV IH];
      if(stop==inf)
        fprintf(fid20,'%s %.6f %.6f %.6f %.8f %.6f %.6f %.6f %.6f %.6f
%.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f
%.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f %.6f\n',
cont0(neweve(j-iline+1),:), odata(j-iline+1,1),odata(j-
iline+1,2),odata(j-iline+1,3),odata(j-iline+1,4),odata(j-
iline+1,5),odata(j-iline+1,6),odata(j-iline+1,7),odata(j-
iline+1,8),odata(j-iline+1,9),odata(j-iline+1,10),odata(j-
iline+1,11),odata(j-iline+1,12),odata(j-iline+1,13),odata(j-
iline+1,14),odata(j-iline+1,15),odata(j-iline+1,16),odata(j-
iline+1,17),odata(j-iline+1,18),odata(j-iline+1,19),odata(j-
```

```
iline+1,20),odata(j-iline+1,21),odata(j-iline+1,22),odata(j-
iline+1,23),odata(j-iline+1,24),odata(j-iline+1,25),odata(j-
iline+1,26),odata(j-iline+1,27),odata(j-iline+1,28),odata(j-
iline+1,29),odata(j-iline+1,30),odata(j-iline+1,31),odata(j-
iline+1,32),odata(j-iline+1,33),odata(j-iline+1,34),odata(j-
iline+1,35),odata(j-iline+1,36));
      end
      if(pok==1)
        set(ctrl.action,'String',[cont0(neweve(j),1:31) ' has been
succesfully computed.']);
        newlines=j+1;
        if(j==length(neweve))
           set(ctrl.pb3,'String','Save last results');
           msgbox('Press "save last results" button to save new
data.','Info','warn');
        else
            msgbox('Press "process" button to follow processing and
save new data.','Info','warn');
        end
        return;
      end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%
newlines=length(neweve)+1;
set(ctrl.action,'String','Program finished');
```

Central_defV2.m

```
function
[SD,PGAu,PGAc,Ia,Trif,Cav1,PGV,Psv,IH]=Central_defV2(FILEs,clnoff,pok,
firstime,direct_name)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Accelerometric data processing             %
%      NERIES NA5 (by M.Tapia and Albert Marsal)        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-----reading accelerograms-----------------------------%
%   Two columns: time (t) and amplitude (a)              %
%--------------------------------------------------------%
Datos=load(FILEs);
t=Datos(:,1);
a=Datos(:,2);
%-------------Despreciate time negative values-----------------------
----%
for i=1:length(t)
    if (t(i)>=0)
        break;
    end
end
t=t(i:length(t));
a=a(i:length(a));
%-------------Signal Duration=last sample of time--------------------
----%
SD=t(length(t));
%-------------------------------------------------------------------------
----%
if(clnoff==1)
    a_cor=offsetcleaner(a,t);
    a=a_cor;
end
```

```
At=t(2)-t(1);
Fnyq=1/(2*At);
%Get the file name from the complet path name
L=length(FILEs);
i=0;
while (FILEs(L-i)~='/')
    i=i+1;
end
file=FILEs(L-i+1:L);
%-----PGA unfiltered (cm/s2)----------------------------%
%-------------------------------------------------------%
PGAu=max(abs(a));
%-----Filtering the accelerogram-----------------------%
%   Butterworth filter from 0.1 to (Fnyq-1)Hz           %
%-------------------------------------------------------%
orden=2;
freqinf=0.1;%Hz
freqsup=Fnyq-1.;%Hz
%------Zero padding at the beginning and at the end of the signal-----
----%
% The length of zero padding is 10% of the length of the original
record
%-------------------------------------------------------------------
----%
zero_pad=length(t)/10;
zero_pad=round(zero_pad);
newlength=length(t)+zero_pad;
for i=1:length(t)
    t2(i)=t(i);
end
for i=length(t)+1:newlength
    t2(i)=t2(i-1)+At;
end
for i=1:round(zero_pad/2)
    a2(i)=0;
end
for i=round(zero_pad/2)+1:length(t)+round(zero_pad/2)
    a2(i)=a(i-round(zero_pad/2));
end
for i=length(t)+round(zero_pad/2)+1:newlength
    a2(i)=0;
end
t2=t2';
a2=a2';
%%%%%%-------------------------------------------------------%
[a_filt]=But2polV2(a2,freqinf,freqsup,orden,Fnyq);
%-----PGA filtered (cm/s2)------------------------------%
%-------------------------------------------------------%
PGAc=max(abs(a_filt));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Computed parameters from the accel. time histories%%%%%%

%Arias intensity (cm/s)
Ia=Int_arias(t2,a_filt);
%Trifunac duration (s)
Trif=Trifunac(t2,a_filt);
%CAV, cumulative absolute velocity (cm/s)
Cav1=Cav(t2,a_filt);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-----Finding the velocity and displacement-------------%
```

```
%              in the time domain                        %
%--------------------------------------------------------%
%Velocity (cm/s)
intgr1=cumtrapz(t2,a_filt);
%Displacement (cm)
intgr2=cumtrapz(t2,intgr1);
%----- Option to save aceleration, velocity and displacement --%
if(firstime==1)
    outfile=cat(2,direct_name,'/',file(1:length(file)-3),'avd');
    fid30=fopen(outfile,'w+');
    for i=1:100
        format(i)=' ';
    end
    fprintf(fid30,'Filtered acceleration (cm/s^2)');
    fprintf(fid30,'       Velocity (cm/s)');
    fprintf(fid30,'       Displacement (cm)\n');
    position=ftell(fid30);
    for r=1:length(a_filt)
        fprintf(fid30,'%s',format);
        fseek(fid30,position+10,-1);
        fprintf(fid30,'%f',a_filt(r));
        fseek(fid30,position+39,-1);
        fprintf(fid30,'%f',intgr1(r));
        fseek(fid30,position+61,-1);
        fprintf(fid30,'%f\n',intgr2(r));
        position=ftell(fid30);
    end
    fclose(fid30);
end
%-----PGV (cm/s) ---------------------------------------%
%--------------------------------------------------------%
PGV=max(abs(intgr1));

%-----Pseudospectrum velocity 5% damping----------------%
%--------------------------------------------------------%
%damping
amort=0.05;
% PSV computed at these frequencies
ff=[0.15, 0.19, 0.23, 0.28, 0.34, 0.42, 0.52, 0.64, 0.78, ...
    0.96, 1.18, 1.45, 1.78, 2.19, 2.69, 3.31, 4.06, 4.99, 6.13,
7.53,...
    9.25, 11.37, 13.96, 17.15, 21.07, 25.89, 31.80, 39.07];

w=2*pi*ff;
%PSV computation
[fpsv,Psv]=psv(amort,w,a_filt,t2);
%Housner intensity (cm)---------------------------------%
IH=Housner(fpsv,Psv);
%IH=1; %prova
%--------------------------------------------------------%

%When PGA <= 0.01cm/s^2 and PGV <= 1cm/s PSVs for low frequencies are
not offered---%
if  ((PGAc <= 0.01*981)|(PGV <=1))
    for k=1:5;
       Psv(k)= 99999;
    end
end

%Start
plotting%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if(pok==1)
    figure('Name',file,'NumberTitle','off');
    subplot(2,2,1:2);
    plot(t,a);
    title('Accelerogram data, before two poles Butterworth filter from
0.1Hz to Nyquist
frequency','FontWeight','Bold','Color','b','FontSize',14);
    xlabel('Time (s)','FontWeight','Bold');
    ylabel('Acceleration (cm/s^2)','FontWeight','Bold');
    subplot(2,2,3);
    plot(t2,intgr1);
    title('Velocity','FontWeight','Bold','FontSize',13,'Color','b');
    xlabel('Time (s)','FontWeight','Bold');
    ylabel('Velocity (cm/s)','FontWeight','Bold');
    subplot(2,2,4);
    plot(t2,intgr2);

title('Displacement','FontWeight','Bold','FontSize',13,'Color','b');
    xlabel('Time (s)','FontWeight','Bold');
    ylabel('Displacement (cm)','FontWeight','Bold');
    fig=figneries(file,SD,PGAu,PGAc,Ia,Trif,Cav1,PGV,Psv,IH);
end
```

Cav.m

```
function [CAV]=Cav(t,a)
%Cumulative absolute velocity
CAV=trapz(t,abs(a));
```

Housner.m

```
function [IHousner]=Housner(f,psv)
%integral entre 0.1s y 2.5s de periodo para PSV*periodo diferencial de
%frecuencia
X=1./f(6:21);
Y=psv(6:21);
f=f(6:21);
Y=Y.*X.*X;
IHousner=abs(trapz(f,Y));
```

Int_arias.m

```
function [arias]=Int_arias(t,a)
%Computation of the Arias intensity
%input: time, accelerogram (cm/s2)
a2=a.*a;
arias=(pi/(2*981.))*trapz(t,a2);
```

Psv.m

```
function [freq,PSV]=psv(etha,w,a,t)
%Input time acceleration
%damping-->etha=0.05
%vector freq,w
%computation of psv for the frequencies especified in w.
%Psv is computed using the convolution theorem to solve duhamel int.
%-------------------------------------------------------------------
---
%enlargement of the signal to avoid oscilations for low frequencies
%(f~0.05Hz-->Ttot~20s).
```

```
ttot=max(t);
N=length(t);
if ttot < 20;
    t_add=20-ttot;
    counts_add=int32(t_add/(t(2)-t(1)));
    counts_tot=20/(t(2)-t(1));
    %adding zeros at the begining of the signal
    for i=1:counts_add
        aa(i)=0;
    end
    for i=1:N
        aa(i+counts_add)=a(i);
    end
    for i=1:counts_tot
        t(i)=(t(2)-t(1))*(i-1);
    end
    a=aa;
end

%-------------Despreciate time negative values-----------------------
-%
for i=1:length(t)
    if (t(i)>=0)
        break;
    end
end
t=t(i:length(t));
a=a(i:length(a));
%-------------------------------------------------------------------
--%
for i=1:28
    func=((exp(-(etha)*(t)*(w(i)))).*(sin((w(i))*(t))));
    PSV(i)=max((conv(a,func)));
end
At=t(2)-t(1);
PSV=PSV*At;
freq=w./(2*pi);
```

Trifunac.m

```
function [trifunac]=TTrifunac(t,a)
%Computation of the Husid diagram and the Trifunac duration,
%tiempo between 0.05% and 95% from Husid diagram
norma=trapz(t,a.*a);
for i=2:length(a);
    a2=a(1:i).*a(1:i);
    t2=t(1:i);
    H(i)=trapz(t2,a2);
end;
H=H/norma;
for i=2:length(a);
    if H(i)<= 0.05;
        t1=t(i);
    end;
    if H(i)<= 0.95;
        t2=t(i);
    end        ;
end;
trifunac=t2-t1;
```

But2polV2.m

```
function [a_filt]=But2polV2(a,freqinf,freqsup,orden,fnyq)
%Input: accelerogram in time domain
%to be filtered
%Frequency pass-band%
Wn=[freqinf,freqsup]./fnyq;

%Construction of a Butterworth filter with order "orden"
%   computation of the coeficients b and a
[bi,ai]=butter(orden,Wn);

%   filtering the time signal
a_filt=filtfilt(bi,ai,a);
```

Offsetcleaner.m

```
function [a_cor]=offsetcleaner(a,t)
At=t(2)-t(1);
p=polyfit(t,a,1);
for i=1:length(a)
        a_cor(i)=a(i)-p(2)-p(1)*(i-1)*At;
end
%first order fit%
%a_cor(i)=a(i)-p(2)-p(1)*(i-1)*At;
%second order fit%
%a_cor(i)=a(i)-p(3)-p(2)*t(i)-p(1)*t(i)^2;
```

Registervisor

Figneries.m

```
function [f]=figneries(name,sd,pgau,pga,ia,tr,cav,pgv,psv,hous)

%-----------Function for graphic PSV representation and parameters
visualization------------%

ff=[0.15, 0.19, 0.23, 0.28, 0.34, 0.42, 0.52, 0.64, 0.78, ...
    0.96, 1.18, 1.45, 1.78, 2.19, 2.69, 3.31, 4.06, 4.99, 6.13,
7.53,...
    9.25, 11.37, 13.96, 17.15, 21.07, 25.89, 31.80, 39.07];

pos=1;
maxpsv=0;
minpsv=99999;
for i=1:28
    if(psv(i)==99999)
        pos=pos+1;
    else
        if(maxpsv < psv(i))
            maxpsv=psv(i);
        end
        if(minpsv > psv(i))
            minpsv=psv(i);
        end
    end
end
```

```
%-------------------------Defect values-------------------------------
----%
maxpsvad=10;
minpsvad=0.001;
ch=1;
posy=1;
posx=0.16;
xlimit=0.01;

%-----------------Grafic parameters definition----------------------
----%
if((maxpsv < 1)&&(minpsv > 0.001))
    maxpsvad=1;
    minpsvad=0.001;
    endid=28;               %text numbers of displacement
    if (pos==1)
        indexa=15;           %text numbers of acceleration
        endia=28;
        indexd=13;
        mult=0.25;
        cw=0.74;
        posx=0.15;
        xlimit=0.01;
        intersecd=0.01;
        interseca=0.3;
        postd=0.04;
        posta=15;
        ch=0.8;
        posy=1.2;
    else
        indexa=13;          %text numbers of acceleration
        endia=27;           %text numbers of acceleration
        indexd=13;          %text numbers of displacement
        mult=0.1;           %texts llegend
        cw=0.7;            %width
        posx=0.16;          %x position
        xlimit=0.1;
        intersecd=0.003;
        interseca=0.6;
        postd=0.4;
        posta=15;
    end
elseif ((maxpsv > 1)&&(maxpsv < 10))
    maxpsvad=10;
    minpsvad=0.001;
    endid=28;
    if (pos==1)
        indexa=10;
        endia=28;
        indexd=9;
        mult=0.2;
        cw=0.7;
        posx=0.16;
        xlimit=0.01;
        intersecd=0.03;
        interseca=0.6;
        postd=0.04;
        posta=15;
    else
        indexa=11;
```

```
        endia=27;
        indexd=13;
        mult=0.9;
        cw=0.57;
        posx=0.23;
        xlimit=0.1;
        intersecd=0.003;
        interseca=0.6;
        postd=0.4;
        posta=15;
    end
elseif ((maxpsv > 10)&&(maxpsv < 100))
    maxpsvad=100;
    minpsvad=0.001;
    endid=28;
    if (pos==1)
        indexa=9;
        endia=28;
        indexd=7;
        mult=0.87;
        cw=0.57;
        posx=0.23;
        xlimit=0.01;
        intersecd=0.03;
        interseca=0.6;
        postd=0.04;
        posta=15;
    else
        indexa=11;
        endia=27;
        indexd=13;
        mult=0.98;
        cw=0.43;
        posx=0.32;
        xlimit=0.1;
        intersecd=0.003;
        interseca=0.6;
        postd=0.4;
        posta=15;
    end
elseif ((maxpsv > 100)&&(maxpsv < 1000))
    maxpsvad=1000;
    minpsvad=0.01;
    endid=25;
    if (pos==1)
        indexa=5;
        endia=26;
        indexd=4;
        mult=0.87;
        cw=0.7;
        posx=0.16;
        xlimit=0.01;
        intersecd=0.3;
        interseca=6;
        postd=0.04;
        posta=15;
    else
        indexa=7;
        endia=23;
        indexd=9;
        mult=0.98;
```

```
        cw=0.43;
        posx=0.32;
        xlimit=0.1;
        intersecd=0.03;
        interseca=6;
        postd=0.4;
        posta=15;
    end
elseif((maxpsv < 1)&&(minpsv < 0.001))
    maxpsvad=1;
    minpsvad=0.0001;
    endid=30;
    if (pos==1)
        indexa=15;
        endia=32;
        indexd=12;
        mult=0.1;
        cw=0.7;
        posx=0.16;
        xlimit=0.01;
        intersecd=0.003;
        interseca=0.06;
        postd=0.04;
        posta=15;
    else
        indexa=13;
        endia=28;
        indexd=14;
        mult=0.1;
        cw=0.57;
        posx=0.23;
        xlimit=0.1;
        intersecd=0.001;
        interseca=0.3;
        postd=0.4;
        posta=15;
    end
end
angle=45;

%------------------------Graphic edition-----------------------------
----%

f=figure;
resolution=get(0,'Monitorposition');
resolution(1)=resolution(1)+50;
resolution(2)=resolution(2)+50;
resolution(4)=700;
resolution(3)=494;
set(f,'Position',resolution);
axs=axes;
plot(ff(pos:28),psv(pos:28),'LineWidth',3);
xlabel('Frequency (Hz)','FontWeight','Bold');
ylabel('Velocity (cm/s)','FontWeight','Bold');
set(axs,'XScale','log','YScale','log','XGrid','on','Ygrid','on','XLim'
,[xlimit 100],'YLim',[minpsvad maxpsvad],'Position',[posx posy*0.23 cw
ch*0.5]','FontWeight','Bold');

%-------------------Parameters visualization-------------------------
----%
```

```
label12=uicontrol('Style','Edit','Units','normalized','Position',[0.06
0.13 0.4 0.021],'String',['Record Duration (s):  '
num2str(sd)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label=uicontrol('Style','Edit','Units','normalized','Position',[0.06
0.1 0.4 0.021],'String',['raw PGA (cm/s^2):  '
num2str(pgau)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label2=uicontrol('Style','Edit','Units','normalized','Position',[0.06
0.05 0.4 0.021],'String',['PGA (cm/s^2):  '
num2str(pga)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label4=uicontrol('Style','Edit','Units','normalized','Position',[0.06
0.02 0.4 0.021],'String',['Arias intensity  (cm/s):  '
num2str(ia)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label6=uicontrol('Style','Edit','Units','normalized','Position',[0.55
0.13 0.4 0.021],'String',['Trifunac duration (s):  '
num2str(tr)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label8=uicontrol('Style','Edit','Units','normalized','Position',[0.55
0.1 0.4 0.021],'String',['CAV (cm/s):  '
num2str(cav)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label10=uicontrol('Style','Edit','Units','normalized','Position',[0.55
0.05 0.4 0.021],'String',['PGV (cm/s):  '
num2str(pgv)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');
label14=uicontrol('Style','Edit','Units','normalized','Position',[0.55
0.02 0.4 0.021],'String',['Housner intensity (cm):  '
num2str(hous)],'FontSize',8,'BackgroundColor',[1 1
1],'FontName','Arial','FontWeight','bold');

vd=[10000 6000 3000 1000 600 300 100 60 30 10 6 3 1 0.6 0.3 0.1 0.06
0.03 0.01 0.006 0.003 0.001 0.0006 0.0003 0.0001 0.00006 0.00003
0.00001 0.000006 0.000003 0.000001 0.0000006 0.0000003 0.0000001];
va=[100000 600000 300000 100000 60000 30000 10000 6000 3000 1000 600
300 100 60 30 10 6 3 1 0.6 0.3 0.1 0.06 0.03 0.01 0.006 0.003 0.001
0.0006 0.0003 0.0001 0.00006 0.00003 0.00001 0.000006 0.000003];
fw=logspace(-2,2,5000);
hold;
for i=1:length(vd)
    vdv=2*pi*vd(i)*fw;
    if(vd(i)==intersecd)
        plot(fw,vdv,'k','LineWidth',2);
        vdr=vdv;
    else
        plot(fw,vdv,'k');
    end
end
for i=1:length(va)
    vda=va(i)./(fw*2*pi);
    if(va(i)==interseca)
        plot(fw,vda,'k','LineWidth',2);
        var=vda;
    else
        plot(fw,vda,'k');
    end
end
```

```
%--------------Plot legends-------------------------------------------
----%

text(postd,maxpsvad-maxpsvad*mult,'Displacement
(cm)','FontWeight','Bold','Rotation',360-angle);
text(posta,maxpsvad-maxpsvad*mult,'Acceleration
(g)','FontWeight','Bold','HorizontalAlignment','Right','Rotation',angl
e);

%---------Text of displacements----------------------------------------
----%

for i=indexd:endid
    vdv=2*pi*vd(i)*fw;
    dif=abs(var(1)-vdv(1));
    ri=1;
    for j=2:length(var)
        if (abs(var(j)-vdv(j))<dif)
            dif=abs(var(j)-vdv(j));
            ri=j;
        end
    end
    if (vd(i)==intersecd)
        text(fw(ri),vdv(ri),['  '
num2str(vd(i))],'FontSize',8,'FontWeight','Bold','VerticalAlignment','
Top','Rotation',angle);
    else
        text(fw(ri),vdv(ri),['  '
num2str(vd(i))],'FontSize',8,'FontWeight','Bold','VerticalAlignment','
Bottom','Rotation',angle);
    end
end

%----------------Text of aceleration-----------------------------------
----%

for i=indexa:endia
    vda=va(i)./(fw*2*pi);
    dif=abs(vdr(1)-vda(1));
    ri=1;
    for j=2:length(vdr)
        if (abs(vdr(j)-vda(j))<dif)
            dif=abs(vdr(j)-vda(j));
            ri=j;
        end
    end
    if (va(i)==interseca)
        text(fw(ri),vda(ri),[num2str(va(i)/1000) '
'],'FontSize',8,'FontWeight','Bold','HorizontalAlignment','Right','Ver
ticalAlignment','Top','Rotation',360-angle);
    else
        text(fw(ri),vda(ri),[num2str(va(i)/1000) '
'],'FontSize',8,'FontWeight','Bold','HorizontalAlignment','Right','Ver
ticalAlignment','Bottom','Rotation',360-angle);
    end
end
```

Figura.m

```
function figura(n_line,paramfile)
```

```
fid=fopen(paramfile,'r');
if(fid==-1)
    errordlg('Error opening the parameters file','File error');
    return;
end

L=length(paramfile);
i=0;
while ((paramfile(L-i)~='/')&&(paramfile(L-i)~='\'))
    i=i+1;
end
pathname=paramfile(1:L-i);

%Situate on the rigth line
for i=1:(n_line-1)
    tline=fgets(fid);
end

name=fscanf(fid,'%s',1);
sd=fscanf(fid,'%f',1);
pgau=fscanf(fid,'%f',1);
pga=fscanf(fid,'%f',1);
ia=fscanf(fid,'%f',1);
tr=fscanf(fid,'%f',1);
cav=fscanf(fid,'%f',1);
pgv=fscanf(fid,'%f',1);
for i=1:28
    psv(i)=fscanf(fid,'%f',1);
end
hous=fscanf(fid,'%f',1);

f=figneries(name,sd,pgau,pga,ia,tr,cav,pgv,psv,hous);

%%%%%%%Plot acceleration time-series in the same graphic

time_file=cat(2,pathname,cat(2,name(1:4),'\'));
time_file=cat(2,time_file,cat(2,name(5:6),'\'));
time_file=cat(2,time_file,cat(2,name(7:8),'\'));
time_file=cat(2,time_file,name);
if (exist(time_file)==0)
    msgbox('Can not find acceleration record','Info','error');
    fclose('all');
    return;
end
Datos=load(time_file);
t=Datos(:,1);
a=Datos(:,2);
axs1=axes;
set(axs1,'Position',[0.16 0.8 0.7 0.15]);
plot(axs1,t,a);
set(axs1,'YLim',[min(a)-1 max(a)+1]);
title(axs1,['Acceleration and PSV 5% --- '
name],'FontWeight','Bold','FontSize',10,'Color','b');
xlabel(axs1,'Time (s)','FontSize',7,'FontWeight','Bold');
ylabel(axs1,'Acceleration (cm/s^2)','FontSize',7,'FontWeight','Bold');

name=([pathname name(1:8) '-' name(10:15) '-' name(17:24) '-'
name(26:27) '-' name(29:31) '.jpg']);
set(f,'PaperPositionMode','auto')
```

```
print('-f','-djpeg','-r250',name);
close(f);
fclose(fid);
```

pathdef.m

```
function p = pathdef
%PATHDEF Search path defaults.
%   PATHDEF returns a string that can be used as input to MATLABPATH
%   in order to set the path.


%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 1.4.2.1 $ $Date: 2003/01/16 12:51:34 $


% DO NOT MODIFY THIS FILE.  IT IS AN AUTOGENERATED FILE.
% EDITING MAY CAUSE THE FILE TO BECOME UNREADABLE TO
% THE PATHTOOL AND THE INSTALLER.

p = [...
%%% BEGIN ENTRIES %%%
     'E:\VERSIO2\Matlab;', ...
     matlabroot,'\toolbox\matlab\general;', ...
     matlabroot,'\toolbox\matlab\ops;', ...
     matlabroot,'\toolbox\matlab\lang;', ...
     matlabroot,'\toolbox\matlab\elmat;', ...
     matlabroot,'\toolbox\matlab\elfun;', ...
     matlabroot,'\toolbox\matlab\specfun;', ...
     matlabroot,'\toolbox\matlab\matfun;', ...
     matlabroot,'\toolbox\matlab\datafun;', ...
     matlabroot,'\toolbox\matlab\polyfun;', ...
     matlabroot,'\toolbox\matlab\funfun;', ...
     matlabroot,'\toolbox\matlab\sparfun;', ...
     matlabroot,'\toolbox\matlab\scribe;', ...
     matlabroot,'\toolbox\matlab\graph2d;', ...
     matlabroot,'\toolbox\matlab\graph3d;', ...
     matlabroot,'\toolbox\matlab\specgraph;', ...
     matlabroot,'\toolbox\matlab\graphics;', ...
     matlabroot,'\toolbox\matlab\uitools;', ...
     matlabroot,'\toolbox\matlab\strfun;', ...
     matlabroot,'\toolbox\matlab\imagesci;', ...
     matlabroot,'\toolbox\matlab\iofun;', ...
     matlabroot,'\toolbox\matlab\audiovideo;', ...
     matlabroot,'\toolbox\matlab\timefun;', ...
     matlabroot,'\toolbox\matlab\datatypes;', ...
     matlabroot,'\toolbox\matlab\verctrl;', ...
     matlabroot,'\toolbox\matlab\codetools;', ...
     matlabroot,'\toolbox\matlab\helptools;', ...
     matlabroot,'\toolbox\matlab\winfun;', ...
     matlabroot,'\toolbox\matlab\demos;', ...
     matlabroot,'\toolbox\matlab\timeseries;', ...
     matlabroot,'\toolbox\matlab\hds;', ...
     matlabroot,'\toolbox\matlab\guide;', ...
     matlabroot,'\toolbox\matlab\plottools;', ...
     matlabroot,'\toolbox\local;', ...
     matlabroot,'\toolbox\shared\controllib;', ...
     matlabroot,'\toolbox\compiler;', ...
     matlabroot,'\toolbox\signal\signal;', ...
     matlabroot,'\toolbox\signal\sigtools;', ...
```

```
     matlabroot,'\toolbox\signal\sptoolgui;', ...
     matlabroot,'\toolbox\signal\sigdemos;', ...
     matlabroot,'\toolbox\shared\spcuilib;', ...
     matlabroot,'\toolbox\shared\dastudio;', ...
     matlabroot,'\work;', ...
%%% END ENTRIES %%%
     ...
];

p = [userpath,p];
```

read_param.m

```
function p = pathdef
%PATHDEF Search path defaults.
%   PATHDEF returns a string that can be used as input to MATLABPATH
%   in order to set the path.


%   Copyright 1984-2002 The MathWorks, Inc.
%   $Revision: 1.4.2.1 $ $Date: 2003/01/16 12:51:34 $


% DO NOT MODIFY THIS FILE.  IT IS AN AUTOGENERATED FILE.
% EDITING MAY CAUSE THE FILE TO BECOME UNREADABLE TO
% THE PATHTOOL AND THE INSTALLER.

p = [...
%%% BEGIN ENTRIES %%%
     'E:\VERSIO2\Matlab;', ...
     matlabroot,'\toolbox\matlab\general;', ...
     matlabroot,'\toolbox\matlab\ops;', ...
     matlabroot,'\toolbox\matlab\lang;', ...
     matlabroot,'\toolbox\matlab\elmat;', ...
     matlabroot,'\toolbox\matlab\elfun;', ...
     matlabroot,'\toolbox\matlab\specfun;', ...
     matlabroot,'\toolbox\matlab\matfun;', ...
     matlabroot,'\toolbox\matlab\datafun;', ...
     matlabroot,'\toolbox\matlab\polyfun;', ...
     matlabroot,'\toolbox\matlab\funfun;', ...
     matlabroot,'\toolbox\matlab\sparfun;', ...
     matlabroot,'\toolbox\matlab\scribe;', ...
     matlabroot,'\toolbox\matlab\graph2d;', ...
     matlabroot,'\toolbox\matlab\graph3d;', ...
     matlabroot,'\toolbox\matlab\specgraph;', ...
     matlabroot,'\toolbox\matlab\graphics;', ...
     matlabroot,'\toolbox\matlab\uitools;', ...
     matlabroot,'\toolbox\matlab\strfun;', ...
     matlabroot,'\toolbox\matlab\imagesci;', ...
     matlabroot,'\toolbox\matlab\iofun;', ...
     matlabroot,'\toolbox\matlab\audiovideo;', ...
     matlabroot,'\toolbox\matlab\timefun;', ...
     matlabroot,'\toolbox\matlab\datatypes;', ...
     matlabroot,'\toolbox\matlab\verctrl;', ...
     matlabroot,'\toolbox\matlab\codetools;', ...
     matlabroot,'\toolbox\matlab\helptools;', ...
     matlabroot,'\toolbox\matlab\winfun;', ...
     matlabroot,'\toolbox\matlab\demos;', ...
     matlabroot,'\toolbox\matlab\timeseries;', ...
     matlabroot,'\toolbox\matlab\hds;', ...
```

```
     matlabroot,'\toolbox\matlab\guide;', ...
     matlabroot,'\toolbox\matlab\plottools;', ...
     matlabroot,'\toolbox\local;', ...
     matlabroot,'\toolbox\shared\controllib;', ...
     matlabroot,'\toolbox\compiler;', ...
     matlabroot,'\toolbox\signal\signal;', ...
     matlabroot,'\toolbox\signal\sigtools;', ...
     matlabroot,'\toolbox\signal\sptoolgui;', ...
     matlabroot,'\toolbox\signal\sigdemos;', ...
     matlabroot,'\toolbox\shared\spcuilib;', ...
     matlabroot,'\toolbox\shared\dastudio;', ...
     matlabroot,'\work;', ...
%%% END ENTRIES %%%
     ...
];

p = [userpath,p];
```

registervisorV9.m

```
function registervisorV9(action)

global ctrl;

if(nargin<1)
    f=openfig('Regvisor.fig');
    ctrl=guihandles(f);
    return
end
if(nargin==1)
    if (strcmp(action,'open'))
        read_param(ctrl);
    end
    if (strcmp(action,'view'))
        n_line=get(ctrl.lb,'Value');
        paramfile=get(ctrl.text,'String');
        for i=1:length(n_line)
            figura(n_line(i),paramfile);
        end
    end
    if (strcmp(action,'close'))
        close('all');
    end
end
```

Regvisor.m

```
function varargout = Regvisor(varargin)
% REGVISOR M-file for Regvisor.fig
%      REGVISOR, by itself, creates a new REGVISOR or raises the
existing
%      singleton*.
%
%      H = REGVISOR returns the handle to a new REGVISOR or the handle
to
%      the existing singleton*.
%
%      REGVISOR('CALLBACK',hObject,eventData,handles,...) calls the
local
```

```
%      function named CALLBACK in REGVISOR.M with the given input
arguments.
%
%      REGVISOR('Property','Value',...) creates a new REGVISOR or
raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before Regvisor_OpeningFunction gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Regvisor_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Regvisor

% Last Modified by GUIDE v2.5 04-Oct-2007 12:50:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Regvisor_OpeningFcn, ...
                   'gui_OutputFcn',  @Regvisor_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Regvisor is made visible.
function Regvisor_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Regvisor (see VARARGIN)

% Choose default command line output for Regvisor
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Regvisor wait for user response (see UIRESUME)
% uiwait(handles.fig);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = Regvisor_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in lb.
function lb_Callback(hObject, eventdata, handles)
% hObject    handle to lb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns lb contents as cell
array
%        contents{get(hObject,'Value')} returns selected item from lb


% --- Executes during object creation, after setting all properties.
function lb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pb1.
function pb1_Callback(hObject, eventdata, handles)
% hObject    handle to pb1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pb2.
function pb2_Callback(hObject, eventdata, handles)
% hObject    handle to pb2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pb3.
function pb3_Callback(hObject, eventdata, handles)
% hObject    handle to pb3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Appendix B: User manual for the software of seismic data calculation within the Neries project: Paramacc

User manual for the software of seismic data calculation within the Neries project – February 2010

1-Installing the software

This application can be used under Windows and Unix platforms for users who have and do not have MATLAB installed. Therefore we can distinguish 4 types of users:

-Windows with Matlab.
-Windows without Matlab.
-Unix with Matlab.

1.1-Windows platform with MATLAB

If your PC uses Windows and you are going to run this application with MATLAB, copy all files *.m, *.fig attached in the zipped file paramaccV7_Matlabs.zip to your working directory. Note that if your working directory is not the same that MATLAB uses as default ("C:\Program Files\MATLAB*Version*\work"), you must change the MATLAB path with the option "**Current Directory**" in the menu toolbar. To run the application type the following at the MATLAB command prompt:

>> paramaccV9

1.2-Windows platform without MATLAB

Install the MCR (MATLAB Component Runtime) by running the MCRinstaller.exe placed in the folder \MCR. This will allow you to run the application without MATLAB. Copy *.CTF and *.exe of the zipped file paramaccV9_compiled.zip to your working directory. If you are not a Windows XP user, add the following directory to your system path:

C:\<mcr_*root*>*version*\runtime\win32

Run the application double clicking in the file **paramaccV9.exe**.

1.3-Unix platform with MATLAB

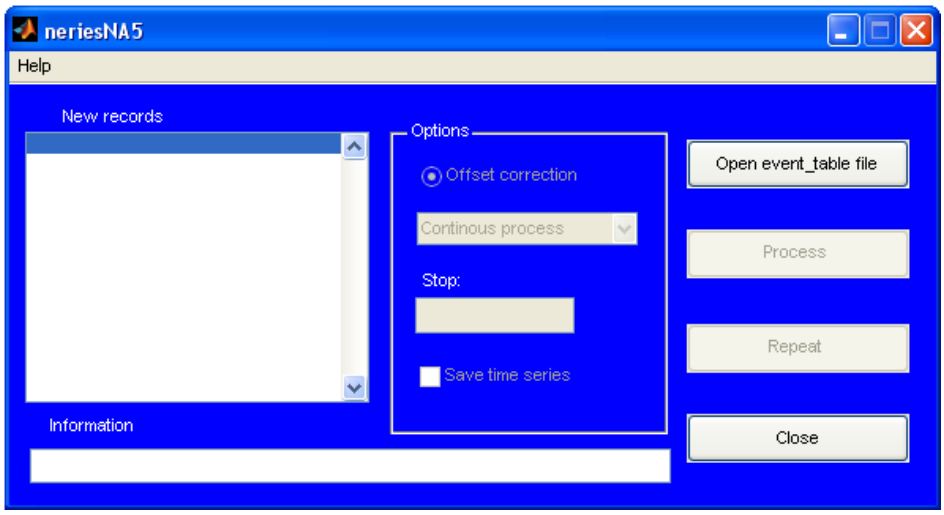
Follow the same instructions done in the paragraph 1.1.

2-User manual

This application allows the calculation of accelerograms' parameters according to the agreements taken in the Neries project. These records are registered in an input file and with a specific format that makes impossible to use another file format to run the program. As a result of the calculation process, an output file will be generated with the parameters for each record. This last file is the one required for the visualization application *(registervisor)*.

2.1-The window

When executing the application a window like this is displayed:



The window has the following elements: four action buttons (right side), a box with a list of new records, a text box to display additional information and an option box with a pop-up menu, a selection button, a check button and a small text box.

The list box contains all the new processable records and the text box below ("Information") displays the work in process. The two boxes to the left are only for showing process information to the user.

The different buttons start the following actions:

-The "Open event_table file" button opens a browse dialog to choose the file *.m, *.dat or *.txt which contains data related to the input records. This file will usually have the standard name "events_table.txt".

-The "Process" button starts the parameters computation according to the options chosen in the "Options" box. The "Options" box allows the user to:

-Make an offset correction of the original record to improve the output results when the button "Offset correction" is selected.

-Select a continuous process or a process with interruptions with the pop-up menu. The parameter data and the intermediate time-histories (acceleration, velocity and displacement) calculated for the last record will de displayed at each stop and the user will be able to check them in

a screen plot. The number of records processed between stops is chosen in the “Stop” text box below.

-Save intermediate time-histories (acceleration, velocity and displacement) with “Save time series”.

-The “Repeat” button recomputes the records that have been processed before with the button “Process” when the user doesn’t agree with results and wants to change any option. This function is only available when a process with stops is chosen.

-The “Close” button ends the application.

2.2-Instructions

For a normal execution of the application follow the instructions listed below:

-Click “Open event_table file” to choose your input event file. In the same folder should be located the “parameters_table.txt”. If there is no parameters file, a new “parameters_table.txt” will be created and it will be supposed that all the records are computed for the first time.
When a file “parameters_table.txt” exists in the folder, the application will add to the file the results for new records not processed before. The application compares the records in the input file “events_table.txt” and the ones in “parameters_table.txt”.
The new records should be placed at the end of the events file “events_table.txt” to make a more understandable process.

-The window “New records” displays the new records to process, which are the ones not found in the “parameters_table.txt”. Before processing you can choose to:

-Apply an offset correction to the new record.

-Process all the new records at once.

-Process a number of records entered in the text box “Stop” with the possibility of checking the last one.

-Save time-histories of each record processed in a folder chosen by the user.

-Click “Process” to start processing according to the options before.

-If a continuous process has been chosen, the application will compute all new records adding the new calculated parameters at the end of the “parameters_table.txt”. Offset correction will be applied if chosen and each record’s time-histories will be saved in a new file if “Save time series” had also been chosen.

-If a stop while processing has been chosen (“Stop each…”) you must fill the text box “Stop” with the number of records processed between interruptions. Offset correction will be applied to this number of records if chosen and time-histories will be saved if “Save time series” is activated.

Once the application stops to display the parameters and time-histories, you can change the options. At this point two choices exist:

-If you agree with previous results, click “Process” to save new calculated parameters and to restart computing next records. Time-histories will have been saved before clicking “Process” if “Save time series” was activated.

-If you don’t agree with previous results displayed during the stop, you may change the options (for example, apply offset correction if it wasn’t applied) and then click “Repeat” to reprocess the parameters of all records until last stop. If the option “Stop each…” is still selected with the same number in “Stop”, the application will stop at the same record and will display the new results. If you agree, click “Process” to save and follow processing according to the options.
New calculated time-histories will overwrite the last ones if “Save time series” was activated and you click on “Repeat” button.

In any stop you can enable or disable the option “Save time series”. Consequently, from that moment the time-histories will start or stop being saved.

-Click “close” to end the application.

Finally, in the menu bar there is the option “Help” that simply opens this document in order to offer the necessary instructions to people who had not initial access to this document.

Appendix C: User manual for the software of seismic data visualization within the Neries project: Registervisor

User manual for the software of seismic data visualization within the Neries project

1-Installing the software

This application can be used under Windows and Unix platforms for users who have and do not have MATLAB installed. Therefore we can distinguish 4 types of users:

-Windows with Matlab.
-Windows without Matlab.
-Unix with Matlab.

1.1-Windows platform with MATLAB

If your PC uses Windows and you are going to run this application with MATLAB, copy all files *.m, *.fig attached in the zipped file registervisor_Matlabs.zip to your working directory. Note that if your working directory is not the same that MATLAB uses as default ("C:\Program Files\MATLAB*Version*\work"), you must change the MATLAB path with the option "**Current Directory**" in the menu toolbar. To run the application type the following at the MATLAB command prompt:

>> registervisorV9

1.2-Windows platform without MATLAB

Install the MCR (MATLAB Component Runtime) by running the MCRinstaller.exe placed in the folder \MCR. This will allow you to run the application without MATLAB. Copy *.CTF and *.exe of the zipped file registervisor_compiled.zip to your working directory. If you are not a Windows XP user, add the following directory to your system path:

C:\<mcr_*root*>*version*\runtime\win32

Run the application double clicking in the file **registervisorV9.exe**.

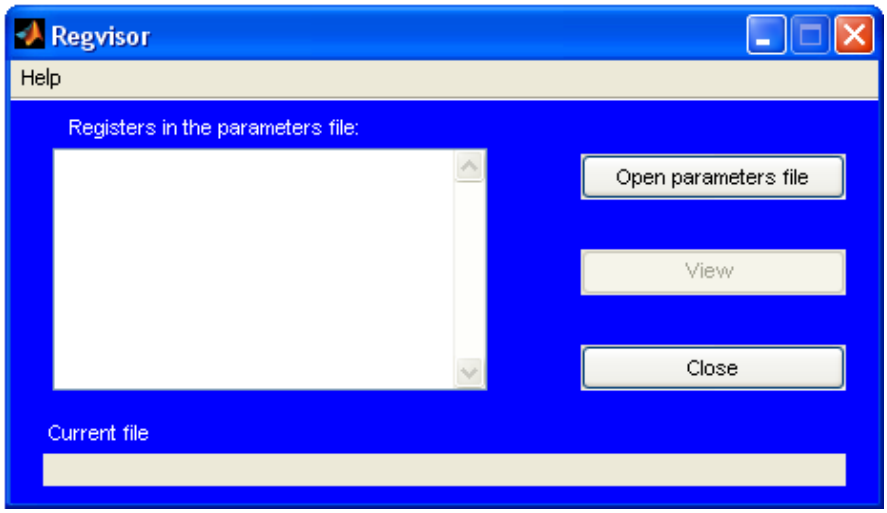
1.3-Unix platform with MATLAB

Follow the same instructions done in the paragraph 1.1.

2-User manual

This application must be only used to display the results of the parameters calculated with Neries software. This means that only the files generated with the Neries software will be read correctly.

A window like this one is shown when the application is executed:



The window contains three buttons, a list box and a smaller text box. The instructions to display the parameters are the following:

-Click the “Open parameters file” button to browse and choose your file “parameters_table.txt” that can be also in *.dat or *.m extension. Remember that the format of input files must be correct and only files generated with the Neries software should be used.

-The program will read all the events included in the file and will show their names in the list box. The text box named “Current file” contains the path of input file.

-Select the records you want to display by highlighting elements in the list window. Click “View” button to display the parameters. This process can be repeated indefinitely.

-For each event displayed an image *event_name*.jpg will be saved in the same folder that the input file. The image shows record duration, PGA, corrected PGA, Arias intensity, Trifunac duration, CAV, PGV, Housner intensity and a PSV plot at 28 frequencies from 0.15 to 39Hz.

-Click “Close” button to end the program.

Finally, in the menu bar there is the option “Help” that simply opens this document in order to offer the necessary instructions to people who had not initial access to this document.